**Application Packaging: A Streamlined Overview**

Application packaging is essential for organizations managing large software portfolios. It simplifies software deployment, ensures standardization, improves security, and reduces support costs. This tutorial shares real-world insights into effective packaging, aiming to help both technical professionals and decision-makers understand the full process.

**Why Application Packaging Matters**

Modern enterprises often run hundreds of applications across diverse environments. Each app has unique installation requirements, making manual deployments impractical and error-prone. Application packaging addresses this by creating standardized, deployable formats like MSI, MSIX, or App-V. These formats enhance compatibility, reduce deployment time, and facilitate lifecycle management.

**Industry Expectations and Trends**

Organizations increasingly expect application packagers to manage the entire process—from discovery to deployment. This consolidation reduces costs and speeds up delivery but demands professionals with comprehensive skills across all packaging stages.

**End-to-End Packaging Process**

1. **Application Discovery**:  
   The first and most critical step. It involves validating source files, confirming app functionality in the target environment, and gathering configuration requirements. Proper discovery prevents issues downstream and may require vendor support or the creation of separate packages for different business unit needs.
2. **Application Packaging**:  
   Using tools and best practices, packagers create installation packages based on discovery data. Depending on organizational standards, different packaging formats and practices may apply.
3. **User Acceptance Testing (UAT)**:  
   Conducted in production-like virtual environments, UAT ensures the package behaves as expected. Testing should reflect real-world use across departments, and ideally be done post-import into tools like Configuration Manager or Intune.
4. **Deployment**:  
   Controlled, phased deployment is vital to mitigate risks. Packages should be thoroughly tested and ideally managed using tools such as Microsoft Endpoint Manager, which supports dependency management, supersedence, and rollback plans.

**Key Recommendations**

* **Use containerization (App-V, MSIX)** for improved compatibility and isolation.
* **Leverage tools** like PacKit to automate and streamline packaging and deployment.
* **Maintain documentation** (discovery and UAT) for future updates or audits.
* **Avoid manual installs**—even for small deployments, as they become difficult to scale or track.
* **Keep environments tidy** with consistent naming conventions and periodic reviews to retire unused applications.
* **Implement Application Rationalization**: Analyze apps regularly to retire or consolidate redundant software, optimizing costs.

**Post-Deployment Best Practices**

Applications must be monitored after deployment. If issues arise, it's better to rework and redeploy the full package rather than patching via separate fixes. Also, use the Windows-as-a-Service (WaaS) model as a prompt for regular testing and updating applications to remain compliant and functional.

**Final Thoughts**

Application packaging is not a one-size-fits-all solution but a critical component of modern IT management. Success requires a thorough understanding of the entire lifecycle—from initial request to retirement. Embracing best practices and automation tools ensures scalable, secure, and cost-effective application management.